Remarks

In the present response, four claims (11, 17, 39, and 47) are amended to correct typographical errors. No new matter or new issues are presented with the amendments.

I. Claim Objections:

The Office Action objected to claims 11, 17, 39, and 47 because of typographical errors. Applicants apologize for the occurrence of these errors. Each of these claims is amended to correct a typographical error. These amendments are merely made to comply with an objection as to form. Furthermore, these amendments merely place the application in a better form for appeal. As such, per 37 CFR 1.116(b), Applicants respectfully ask the Examiner to enter these amendments.

II. Claim Rejections: 35 USC § 103(a):

Claims 1-5, 8, 11, 14, 17, 20, 22-27, 30, 33, 36, 39, 42, 45, and 47 are rejected under 35 USC § 103(a) as being unpatentable over Detlefs in view of USPN 4,920,538 (Chan). Applicants respectfully traverse.

To establish a prima facie case of obviousness, three basic criteria must be met. First, there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art to modify the reference or to combine reference teachings. Second, there must be a reasonable expectation of success. Finally, the prior art cited must teach or suggest all the claim limitations. *See* M.P.E.P. § 2143. Applicants assert that the rejection does not satisfy these criteria.

A. No Suggestion or Motivation to Modify/Combine References

Claims 1, 23, and 47 recite methods and apparatus directed toward a static checker. No motivation, teaching, or suggestion exists for combining Detlef and Chan to teach Applicants' claimed limitations in a static checker.

The Detlef reference is directed toward an "Extended **Static Checker** (called ESC), a programming tool that catches errors at compile time that ordinarily are not caught until runtime." (Emphasis Added. Page 2, lines 5-6). By contrast, Chan is directed toward a **dynamic checker**: "Microcode sequence errors are determined at the completion of each microprogram node execution by comparing a path identifier **generated at runtime** with the global label stored earlier." (Abstract. Emphasis added. See also specification for details on runtime execution).

Static checkers determine errors at compile time without simulating or executing the program. By contrast, dynamic checkers (or runtime checkers) execute or simulate the execution of the computer program to determine an error. Since static and dynamic checkers use widely different methods for checking for errors, one skilled in the art would not be motivated to modify Detlef with Chan to arrive at the claimed invention. In fact, Applicants note in their Background a number of distinct disadvantages with dynamic checkers. (For list of disadvantages, see paragraph [0008] of US 2002/0046393).

B. References Not Teach Claimed Elements

First, the Examiner admits that "Detlef does not expressly disclose the limitation wherein the converting step includes inserting flow control labels into the sub-equation of the logical equation, the flow control labels identifying conditional branch points in the specified computer program." The Examiner states that "Detlefs does show that any sub-equation of the logical equation can be given a label (see page 29, section 6, paragraph 2, lines 1-2)." The Examiner further states, though, that Chan discloses this limitation (see column 1, lines 26-35 and 38-53).

Applicants respectfully disagree. Detlefs (at page 29, section 6, paragraph 2, lines 1-2) teaches a static checker that reports the location of specific error message. In other words, "the label encodes the source position and error type." (Id.). Chan teaches a marker that "is computed **at runtime** and is matched with the stored marker to detect any wild branches." (Emphasis added. Col. 1, lines 41-42). The combination thus teaches (1) reporting the location of a specific error message plus (2) computing a marker at runtime to detect wild branches. In other words, this combination teaches and suggests a dynamic checker that, at runtime, reports the location of specific error messages. This combination, however, does not teach or suggest the claimed limitation. Claim 1 specifically recites "inserting flow control labels into the sub-equations of the logical equation, the flow control labels identifying conditional branch points in the specified computer program." Similar limitations are recited in independent claims 23 and 47.

Second, the Examiner contends that Detlefs teaches a theorem prover with any of the flow control labels for conditional branch points of the program associated with the identified variable values. The Examiner cites Page 23, section 4, paragraph 1, lines 4-8. Applicants have scrutinized this section and find no such teaching. This section is reproduced below:

> The condition is submitted to an automatic theorem-prover, just like in program verification, but unlike in program verification, we have no interest in the case where the theorem prover succeeds. Instead, the tool post-processes theorem-prover failures into meaningful error messages.

In short, this section plainly states that a condition is submitted to a theorem-prover. The program then processes failures into error messages. Detlefs later discusses that an exact location of these error messages is reported. (See page 29, section 6). This section, however, does not teach or suggest "flow control labels for conditional branch points of the program associated with the identified variable values." Similar limitations are recited in independent claim 23.

Third, the Examiner states that Detlefs teaches an error message that includes a **program trace that identifies a path through the computer program** when the counter-example identifies one or more of the flow control labels. The Examiner cites Detlef (Page 29, section 6, paragraph 2, lines 3-4 and 7-9) and Chan (Col. 1, lines 26-35).

This section of Detlef is reproduced below:

> If the prover finds a counterexample, it emits the set of labels of relevant subformulas that are false in the counterexample. The name of the label encodes the source position and error type. This makes it straightforward to translate failed proofs into specific error messages.

This section of Chan is reproduced below:

> One of the problems that arises in trying to check entire microcode sequences is checking the execution of conditional branching instructions. This is because branches outside the path taken in calculating the original check key will result in different check keys being calculated for each branch path.
>
> It is an object of the present invention to provide a microprogrammed controller that is self-verifying in which valid branches along the execution path are verified and invalid branches are detected and isolated.

Applicants submit that Chan teaches a controller that is self-verifying. Specifically, valid branches along the execution path are verified. Invalid branches along the execution path are detected and isolated. These sections, however, do not teach or suggest (alone or in combination) the claimed recitations. Claim 1 specifically recites: "program trace **that identifies a path through the computer program** when the counter-example identifies one or more of the flow control labels." Similar limitations are recited in claims 23 and 47.

C. Conclusion Section II:

For at least the foregoing reasons, claims 1, 23, and 47 are patentable in view of Detlefs and Chan. Applicants respectfully request withdrawal of the rejection of claims 1, 23, and 47.

Claims 2-4, 8, 11, 14, 17, 20, 22 depend from claim 1, which is believed to be allowable. Therefore, these claims are also believed to be allowable for at least the same reasons as claim 1. Withdrawal of the rejection of claims 2-4, 8, 11, 14, 17, 20, 22 is respectfully requested.

Claims 24-27, 30, 33, 36, 39, 42, and 45 depend from claim 23, which is believed to be allowable. Therefore, these claims are also believed to be allowable for at least the same reasons as claim 23. Withdrawal of the rejection of claims 24-27, 30, 33, 36, 39, 42, and 45 is respectfully requested.

III. Claim Rejections: 35 USC § 103(a):

Claims 6, 7, 9, 10, 12, 13, 15, 16, 18, 19, 21, 28, 29, 31, 32, 34, 35, 37, 38, 40, 41, 43, 44, and 46 are rejected under 35 USC § 103(a) as being unpatentable over Detlefs in view of USPN 4,920,538 (Chan) as applied to claims 1-5, 8, 11, 14, 17, 20, 22-27, 30, 33, 36, 39, 42, and 45 above, and further in view of USPN 5, 854,924 to Rickel et a. (hereinafter Rickel).

References Not Teach Claimed Elements

Applicant acknowledges that Rickel is generally directed to a debugging tool for debugging a binary program file. (See Abstract). Rickel, however, does not cure the deficiencies of Detlefs and Chan. As such, Applicants reiterate the arguments above with respect to independent claims 1, 23, and 47 and all corresponding dependent claims as articulated in Section II.

CONCLUSION

In view of the above, Applicants believe claims 1-47 are in condition for allowance. Allowance of these claims is respectfully requested.

Any inquiry regarding this Amendment and Response should be directed to Philip S. Lyren at Telephone No. (281) 514-8236, Facsimile No. (281) 514-8332. In addition, all correspondence should continue to be directed to the following address:

Hewlett-Packard Company
Intellectual Property Administration
P.O. Box 272400
Fort Collins, Colorado 80527-2400

Respectfully submitted,

[illegible]

Philip S. Lyren
Reg. No. 40,709
Ph: 281-514-8236

CERTIFICATE UNDER 37 C.F.R. 1.8: The undersigned hereby certifies that this paper or papers, as described herein, are being deposited in the United States Postal Service, as first class mail, in an envelope address to: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450 on this 24th day of June, 2004.

By Be Henry
Name: Be Henry